Language Learning Diary - Part Five

Linas Vepštas

Jan 2022 - March 2022

Abstract

The language-learning effort involves research and software development to implement the ideas concerning unsupervised learning of grammar, syntax and semantics from corpora. This document contains supplementary notes and a loosely-organized semi-chronological diary of results. The notes here might not always makes sense; they are a short-hand for my own benefit, rather than aimed at you, dear reader!

Introduction

Part Five of the diary on the language-learning effort continues work on the English dataset.

Good progress was made. It appears that we have a high-quality, stable, debugged clustering algorithm in place, and it is generating good-quality clusters. However, dozens of questions and hypothesis arise, which is a bit overwhelming. The lack of any kind of theoretical foundations for any of this stuff is frustrating. Without theory, it is hard to get strong insight.

At any rate, results seem solid enough that we can now move on to the next steps of the program. I'm hoping to start work on multi-sentence and cross-paragraph correlations in the coming months. Although, before doing this, perhaps a comparison to the hand-built LG grammars is in order.

Summary Conclusions

A summary of what is found in this part of the diary:

- From Experiment-10: Ranked-MI (Variational MI) seems to mostly make good cluster recommendations, except when it doesn't. Since the actual merge proceeds via a Jaccard-type voting in of disjuncts, it is possible that the poor recommendations results in few contributions. But there is no obvious way to automate a quality measurement.

- From Experiment-10: A list of the first 31 clusters is presented. It seems OK but not great. Hard to say, as the Jaccard-selection mechanism selects membership, and we don't have a window into that. Is there a way of visualizing the "word-sense" of a cluster?

- From Experiment-10: A fairly large number of merges are expanding or merging prior classes. This suggests that "preferential attachment" is hard at work, and so we should not be surprised to later observe scale-free results.
- This implies that ranked-MI/variational-MI automatically, inherently makes these kind of preferential-attachment suggestions. How does this work? What are the details? Are there alternatives?
- As clustering proceeds, there are several dozen different measurements and indicator values that can be tracked, including the dataset entropy, the MMT-Q value, the size of clusters, the ranked-MI of the top-ranked word-pair, and so on. Many graphs of these indicators are presented. Experiment-10 tracks four of these; experiments 11 and 12 tracks dozens of them. A significant portion of the parameter space is explored.
- Graphs of the word-disjunct marginal entropy distribution is presented. This for the dataset, before clustering is started. It appears to be a normal (Gaussian) distribution.
- Multiple alternatives to the variational-MI (ranked-MI) are proposed and developed. These are computed for the top-200 most common words in the dataset (*i.e.* for $N(N+1)/2$ word-pairs, for $N = 200$.) Although they make different ranking recommendations, the overall distributions are nearly identical. Although roughly Gaussian, these have a very fat tail to the left (*i.e.* on the negative side: they find a lot of dis-similarity.) The fat tail is hypothesized to be driven by the number of distinct word-senses (*i.e.* word-senses should be dis-similar.) All of these alternatives are more computationally intensive than the ranked-MI, and thus are not really practical to apply. It is hypothesized that all of these might still end up selecting the same in-group (*i.e.* although the top-ranked pairs differ, the in-groups would not.)
- From Experiment-11: The top ranked-MI pair is used to form the initial seed of a cluster, and then the regular MI is used to nominate additional members. When the Jaccard overlap of the nominees is computed, it is surprisingly low. Why? Is it because MI is pair-wise, and Jaccard is group-wise?
- Despite the low overlap, the majority of sections get merged, only because their counts are below the noise-floor. How do low-count disjuncts influence the MI?
- Setting the noise-floor causes disjuncts with a count below this floor to be swept up into the group. Later, during the preferential attachment and growth of groups, is it possible that this "random noise" hijacks the initial cluster, twisting it in a different direction from where it started? Is it causing the signal to be washed out? What happens if we set the noise-floor to zero?

- After running the algo for about 1060 steps, about 530 word-classes were created (the fact that its about a half is unexplained, but gives a hint of the role of preferential attachment). When comparing the $N(N-1)/2$ class-pairs (for $N = 530$), about 93% of them have a similarity of zero: that is, they are completely orthogonal. That's a good thing: we wanted to have classes that are distinct from one-another, and that is what we are getting. Of the remaining 7%, the distribution of MI is almost perfectly a Gaussian, centered at about MI of negative 3, and an stddev of 3.5. Negative MI is also good: it means that even if they are not perfectly orthogonal, they are almost-so.
- The above stands in contrast to the word-word orthogonality of the remaining (as yet unmerged) words: more than 80% of them have a non-zero similarity. The distribution of these is almost Gaussian, with a bit of a fat tail towards the low-end. Compared to the pre-merge MI distribution, there is little change, except perhaps in the fat tail.

That pretty much wraps things up for this chapter. Merge experiments are ongoing, and further results will be presented in later diary chapters. I don't expect many changes from the above; just some longer runs with maybe clearer results. A noise=0 run is clearly called for. Examining orthogonality as a function of time is also interesting, to be reported later.

TODO

A list of things to do, here or later:

- In population genetics, a neutral evolutionary model with a static population size results in a power-law distribution of alleles in the population. Can this process model also explain the power-law (Zipfian, square-root-Zipfian) distributions seen elsewhere? There is a similar set of results in ecology, with regards to species distribution in an ecological niche.

Expt-10 Merge exploration (Jan 2022)

The end of Diary Part Four describes a dataset with several thousand merged words in it. However, that dataset had assorted issues, and recreating it before investing time in characterizing it seems like a good idea. There were numerous (and ongoing) bug-fixes made. Phew. 5 Jan 2022 Looks like the last of the bugs are now fixed. This took two weeks of very tedious debugging. Ouch. Ready to restart, at last.

Now that we are ready, the questions below arise.

Things worth exploring.

What have we actually got? Some questions about the dataset:

- Distribution: size of word-class vs. rank. This was previously examined in "diary part one", page 99, for a different collection of merge algorithms (and earlier, different datasets). Do we still get something similar?
- Above, "size" might mean "number of words in the word-class" or it might mean "number of disjuncts in word-class" or it might mean "number of disjuncts in word-class with count weighting". Oof. It would be nice to have a dashboard for this, instead of lots of manual work.
- Distribution of word-senses. That is, how many words participate in more than one word-class? If they do participate in more than one word-class, what is the weight in each class?
- For words that belong to word-classes, what fraction of their weight remains unassigned to any word-class?
- Distribution of MI of pairs of word-classes. We might hope that this is low, so that different word-classes are different from one another.
- Distribution of self-MI of word-classes. One might hope that this is high, so that the word-classes do not share much in common with other words or word-classes.
- As above, but distribution of MI of pairs consisting of a word-class, and a word.
- Prior to starting the merge, there's an MI between words and disjuncts. I don't recall examining that in detail, before. Then, after the merge, how does this change?

That's a lot of questions. Not clear which ones should be answered first.

Round 43

Done with bug-fixing.

I'm currently using 'r9-sim-200.rdb' for marginals plus the similarities for the top 200 most frequent words. Then run

```
cp -pr r9-sim-200.rdb r10-merge.rdb
guile -l cogserver-gram.scm
(in-group-cluster covr-obj 0.5 0.2 4 200 100)
```

This works great until merge 43 where we get

```
------ Round 43 Next in line: ranked-MI = 5.6267 MI = 5.1464 ('could must would should may will can mi
In-group size=5: 'to' 'could must would should may will can might' 'you' 'I' 'we'
```

So that looks like a bug. Issues:

- Hard to believe that this is the top ranked-MI pair.
  - Is the MI being computed correctly? It almost surely is, but still ... ??

– Should the ranked-MI for a cluster be de-rated, say, by the number of words in the cluster?

– The next 10 highest ranked-MI merges look great! So maybe somehow the ranked MI for clusters is wrong?

– Since merging is via a Jaccard-overlap selection mechanism, perhaps most of the contributing disjuncts in the bad recommendations will be ignored?

Round 88 - Majority voting bug?

Subsequent merges look .. pretty good, except when they don’t, and then they look ugly.

At round 88 it goes nuts: it merges a few disjuncts of “As as” together. Then it merges zero of them into “As as.i”, and then hits an inf loop, because what is left after the merge still has a high MI.

Conclude: the majority-voting scheme left behind (left unmerged) too much; enough that the MI between what is left is still high. Can we redefine the voting procedure to lessen this?

Bug Fixes

There were multiple bugs, difficult to locate. Some cross-section merges were being done wrong. A bug in cog-delete! was erasing data in DB, thus restarts loaded missing/corrupt data. As of 15 Jan all seems to be fixed.

Round 31

After fixes, the first 31 rounds merge the following words. The is with quorum=0.5 commonality=0.2 noise=4.

15	be have see make	OK-ish
N	class	comments
1	+ — “ ” _	OK
2	, ;	OK
3	was is	OK
4	but and that as	OK
5	? . !	OK
6	He It I There	OK, sentence starters
7	" ” , ? . ! what	Expand group N=5
8	He It I There She This	Expansion of group N=6
9	A No He It I There She This The	Expansion of group above!
10	of in to from	OK
11	his the a he I	odd but plausible
12	’ -- but and that as " ” , ? . ! what + — “ ” _	Yuck! Merge N=4,7,1
13	: ###LEFT-WALL### , ; me	Yuck. Expand N=2

15	be have see make	OK-ish
14	has was is had could	Expand N=3
16	must would	OK
17	asked said has was is had could do did	OK-ish
18	was am think	OK-ish
19	:] , .	OK
20	he she	great!
21	are were	great!
22	It This this what	OK
23	Well You ’ -- but and that as " ” , ? . ! what + — “ ” _ you Oh	Yuck. Expand 12
24	“ Oh	Left-overs from above.
25	might should will may	great!
26	much well good little long	good
27	could must would might should will may can shall	great! Merge 16, 25
28	into of in to from	great! Expand 10
29	He She	great!
30	no nothing his the a he I an	Meh. Expand 11
31	on upon	great!

Notable is that a fairly large number of merges are expanding or merging prior classes. This is curious. This suggests that “preferential attachment” is hard at work, and so we should not be surprised to later observe scale-free results.

This implies that ranked-MI/variational-MI automatically, inherently makes these kind of preferential-attachment suggestions. How does this work? What are the details? Are there alternatives?

Regarding the mediocre merge suggestions: The Jaccard-selection mechanism selects membership, and we don’t have a window into that. Its possible that the mediocre merge recommendations work out so that very few disjuncts get merged into the final cluster. Is there a way of visualizing the “word-sense” of a cluster?

Also being tracked, per merge: rows, cols, lcnt, rcnt, size, sparsity, entropy, ranked-mi, mmt-q. These look good, up to round 28, when lcnt=0, size=0 and some divide-by-zero garbage up the other stats! After that point, the lcnt off by a little bit.

At the conclusion of round 27, mmt-q=+inf.0 is already wrong. But this value gets recorded as part of round 28. During extension of round 27, some occasional garbage:

```
MI(‘in‘, ‘into‘) = +inf.0  rank-MI = +nan.0
```

Turns out is was a bad wild-card de-reference that clobbered the wildcard anchor.

Merge Datasets

The following merge parameters are being pursued in the following datasets:

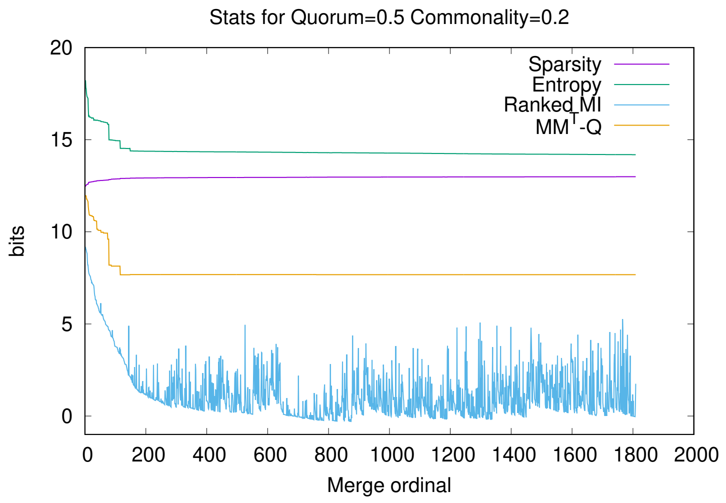
name	quorum	commonality	noise
r10-mrg-q0.5-c0.2.rdb	0.5	0.2	4
r10-mrg-q0.6-c0.3.rdb	0.6	0.3	4
r10-mrg-q0.7-c0.4.rdb	0.7	0.4	4
r10-mrg-q0.8-c0.5.rdb	0.8	0.5	4

Spot sampling of the merges looks much like the above: some pretty good-looking merges interspersed with some head-scratchers. This occasional ugliness continues deeply into the merge stream. What is not clear is how many disjuncts are involved in the uglies, and, if they occur early, exactly how much "noise" they carry away.

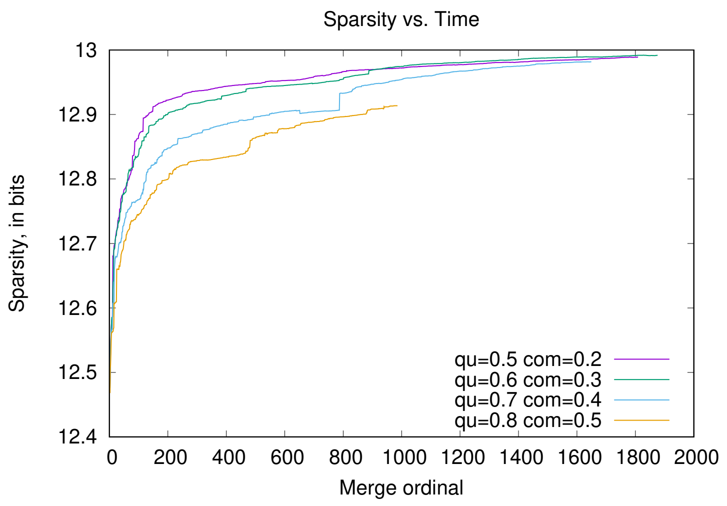
What does come clear from the spot inspection is that ranked-MI is starting to make some pretty poor recommendations. Basically, the clusters get an "unnaturally" large ranked-MI, because they are so large, even though the raw MI to what is being suggested is relatively low. That is, the use of ranked-MI is the reason why the clusters grow preferentially.

For this round, four stats are tracked: the dataset sparsity, the dataset entropy, the ranked MI of the next pair to be selected, and the MM^TQ as defined in the previous diary.

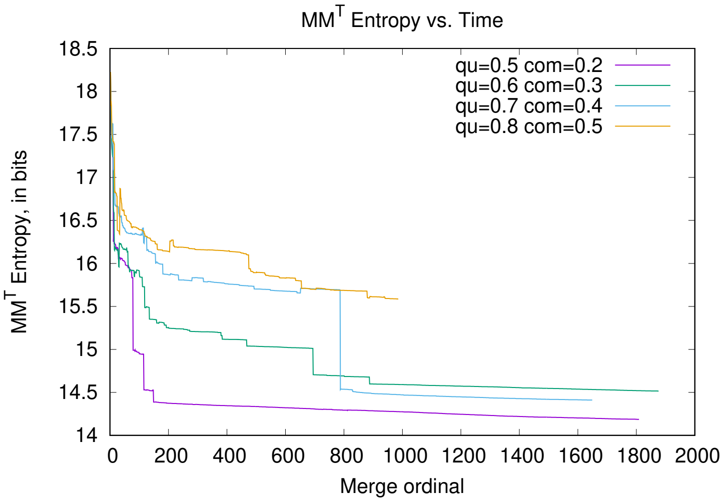
Here's all four of these on one graph:



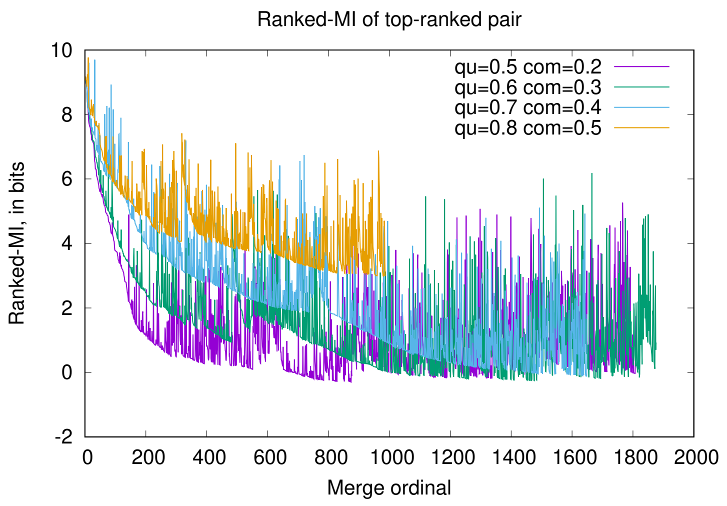
Here's just sparsity, for all four merge parameters. Although it is increasing, it is increasing by very small amounts. The quorum=0.8 run crashed in a way that doesn't make it worth restarting (see newer data below).



The dataset total MM^T entropy. It is dropping because the merges are “bringing order to the chaos”.



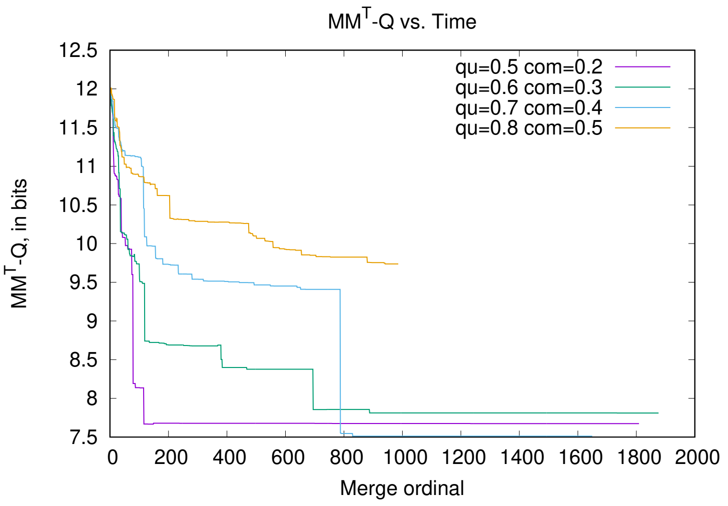
The ranked-MI of the next pair to merge drops, because, naturally, the highest-ranked pairs are merged. It spiked whenever a large cluster is created: the shear size of this cluster guarantees that it will have a large ranked-MI, irrespective of the absolute-MI.



The value Q is being used as a "constant" to offset the common-MI to a non-negative range. The definition was the log of the sum-squared marginals:

$$Q = -\log_2 \sum_d P(*,d) P(*,d)$$

This is "constant" for a fixed dataset, in that it is independent of the word and disjunct. As merges proceed, it changes.



There seems to be a big changes in the quorum=0.7 curve, just before the 800'th merge. Unfortunately, this code failed to track the merged words, so unclear what that's about.

Other issues: the above run fails to compute the marginals correctly, and thus some of the similarities could easily be off. This particular bug is fixed, and a newer run is presented further below.

Notes:

- The quorum=0.8 run died prematurely, after performing 986 merges. At the conclusion, it only had 502 word-classes, so clearly, many classes are being recombined and expanded.
- Failed to write out the MemberLinks on all four datasets, and so these are useless for creating dictionaries.

Marginal Entropy Distributions (Jan 2022)

In the paper "Connector Set Distributions", an extensive examination of various relationships between word distributions, entropy, mutual information and rank is explored. Below, we explore a bit more in that vein, that apparently was not previously explored.

In Diary Part Three we looked at the word-rank vs. support (page 10). Here, we look at rank vs. right-entropy and rank vs. right-MI as well as rank vs log-probability. The goal of asking these questions is two-fold.

- The current merge algo uses ranked-MI to select the next word-pair to anchor a merge. Perhaps some other linear combination of MI and, say, right-entropy might be better?
- The current merge algo uses a majority-voting scheme, by computing an overlap. Perhaps some log-overlap, of some word-disjunct MI would be better?

Anyway, these relationships seem interesting.

Definitions

Some definitions are in order.[1] This repeats definitions given earlier. For a word-disjunct pair (w,d) the frequentist probability is $P(w,d) = N(w,d)/N(*,*)$. Since it has two arguments, this can be called the 'joint probability' or 'joint frequency'.

Total and Marginal Entropies

The total entropy is

$$H = H_{\text{tot}} = -\sum_{w,d} P(w,d) \log_2 P(w,d)$$

The right-marginal entropy for a given word w is

$$h_{\text{right}}(w) = -\sum_{d} P(w,d) \log_2 P(w,d)$$

[1]The definitions here are the same as given in earlier texts. The code that implements these was developed primarily in 2017, and can be found in the 'opencog/matrix' subdirectory of the git repo at AtomSpace. See, in particular, 'frequency.scm' and 'entropy.scm'.

The fractional right-marginal entropy for a given word w is

$$H_{\text{right}}(w) = \frac{h_{\text{right}}(w)}{P(w,*)}$$

The goal of the division by the marginal probability $P(w,*)$ of the word w is to rescale the entropy into a sensible range. For the current dataset, this is in the range of 5 to 25, as graphed below. The $h_{\text{left}}(d)$ and $H_{\text{left}}(d)$ for disjuncts $d$are defined likewise. Note that

$$H_{\text{tot}} = \sum_w P(w,*)\, H_{\text{right}}(w) = \sum_d P(*,d)\, H_{\text{left}}(d)$$

can be used as a consistency check on the numerical results.

The marginal probabilities also allow two more kinds of entropies to be defined. These are the left and right entropies of the left and right marginal probabilities:

$$S_{\text{right}} = \sum_w P(w,*) \log_2 P(w,*)$$

and likewise

$$S_{\text{left}} = \sum_d P(*,d) \log_2 P(*,d)$$

One might argue that the names "left" and "right" are incorrectly interchanged in the above, but this is the naming convention used in the implementation, and changing it risks introducing bugs and confusion.

These two are related to the total mutual information (as defined below) as

$$MI_{\text{tot}} = S_{\text{right}} + S_{\text{left}} - H_{\text{tot}}$$

Total and Marginal Mutual Information

The mutual information for a word-disjunct pair (w,d) is

$$mi(w,d) = P(w,d) \log_2 \frac{P(w,d)}{P(*,d)\, P(w,*)}$$

As always, the fractional MI is

$$MI(w,d) = \frac{mi(w,d)}{P(w,d)}$$

thus scaling it into a sensible range of -5 to +20, as shown below. The marginals proceed likewise, as above, so that the right marginal mi for word w is

$$mi_{\text{right}}(w) = mi(w,*) = \sum_d mi(w,d)$$

and the fractional right marginal mutual information is

$$MI_{\text{right}}(w) = \frac{mi_{\text{right}}(w)}{P(w,*)}$$

The grand-total MI for the dataset is

$$MI = MI_{\text{tot}} = \sum_{w,d} mi(w,d) = \sum_{w} P(w,*)\, MI_{\text{right}}(w) = \sum_{d} P(*,d)\, MI_{\text{left}}(d)$$

As noted earlier, the mutual information is minus the total entropy, up to entropies of the marginal probabilities:

$$MI_{\text{tot}} = S_{\text{right}} + S_{\text{left}} - H_{\text{tot}}$$

Dataset Summary

Starting point: the dataset ‘r9-sim-200.rdb‘ after running ‘compute-mi‘ on it, to get the full collection of stats on (w,d) pairs. The stat summary is:

dataset	H_{tot}	S_{left}	S_{right}	MI_{tot}
r9-sim-200+mi.rdb	19.463	16.535	8.5148	5.5865

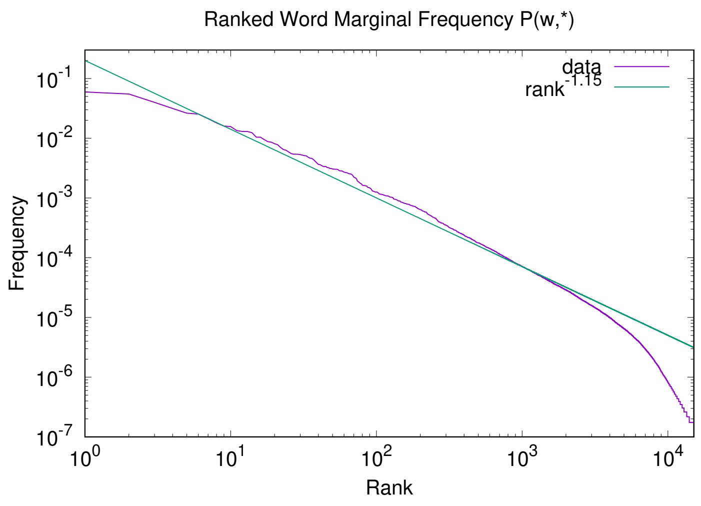
dataset	N_{words}	N_{dj}	N_{pairs}	$N(*,*)$
r9-sim-200+mi.rdb	15083	1043583	2777968	22942644

Here, N_{pairs} is the total number of pairs (w,d) that were observed (i.e. pairs with non-zero counts). The total number of observations of these pairs is $N(*,*)$. The sparsity is $-\log_2 N_{\text{pairs}} / \left(N_{\text{words}} \times N_{\text{dj}}\right) = 12.468$.

Zipf Distribution

Some graphs follow.[2] We begin with a very conventional graph, and a very conventional result. This shows the marginal frequency $P(w,*)$ as a function of the rank. The distribution is almost Zipfian, with a classical Zipf slope. It’s a bit noisier than hoped for; the noise may be due to the somewhat imperfect dataset (it contains a lot of words with escaped backslashes in them, due to a very early parsing bug.)

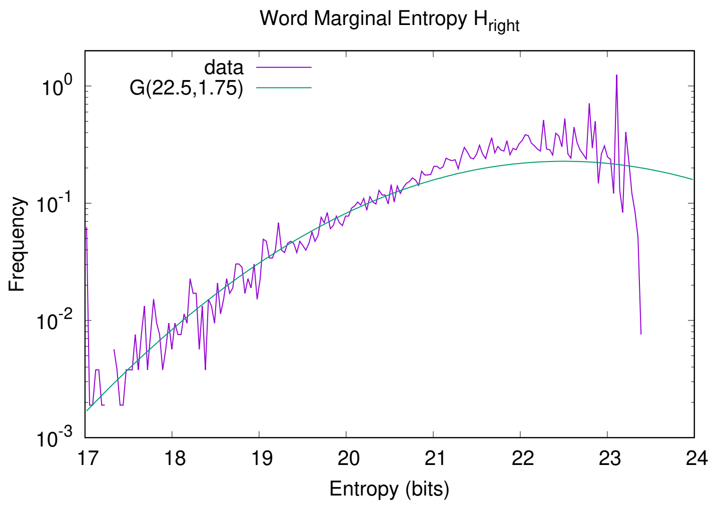
[2]The data was processed with the ‘utils/entropy-marginals.scm‘ scripts in this directory.



A similar figure can be drawn for the entropy; it is very nearly identical.

Entropy Distributions

Next: a histogram of $H_{\text{right}}(w)$ for the 15083 words w in the dataset. These are placed into 200 bins and counted with a weight of 1.0 per word.

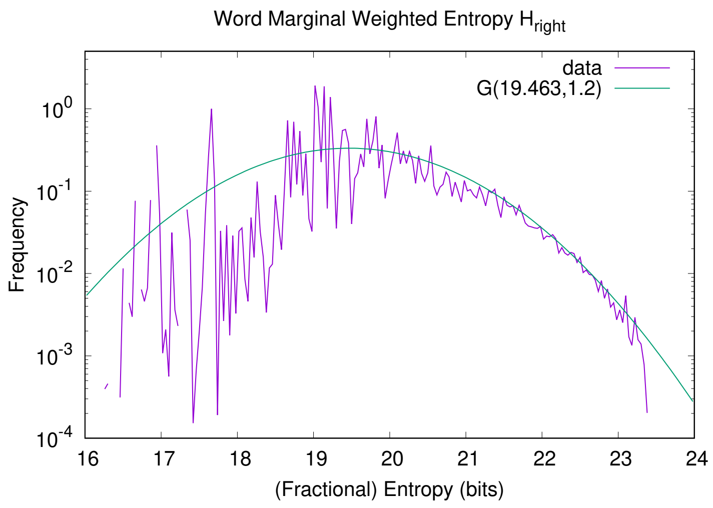


The smooth curve marked G is the Gaussian distribution

$$G(x;\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\exp-\frac{(x-\mu)^2}{2\sigma^2}$$

This fit is not particularly convincing; there's a sharp drop-off to the right that is not being fit. The data is fairly noisy. My guess is that the spiky bits on the right are from infrequently-observed word-disjunct pairs.

This suggests redoing the bin-counts, weighting each word by it's probability. This is shown below.

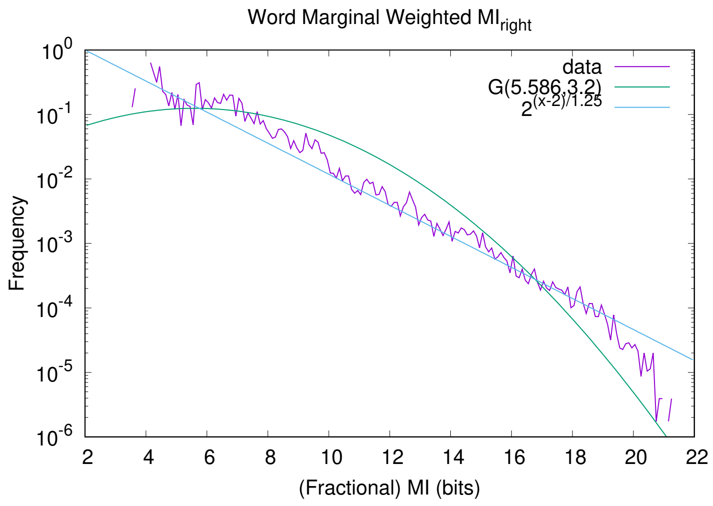


This shows the same data, with each word weighted by the probability of that word. That is, this is a graph of $H_{\text{right}}(w)$ with each point weighted by $P(w,*)$. This makes it clear that the right side of the picture, the high entropy side, is associated with infrequent words.

Also, this time, the Gaussian is an excellent fit. Note that the Gaussian is centered on $H_{\text{tot}} = 19.463$ for the dataset. This is as it should be: the total entropy is the average of this distribution.

MI Distributions

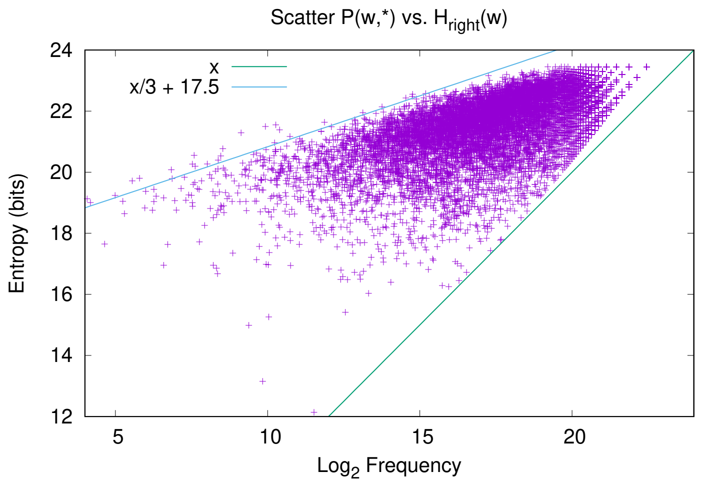
Same as above, but for the word-disjunct mutual information.



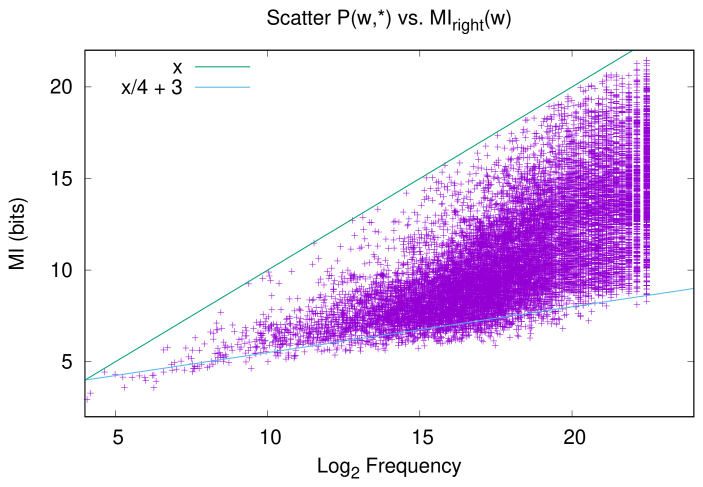
It seems reasonable to expect a Gaussian distribution here, also, but it appears to not be the case. A log-linear fit seems better. It's not clear whether this should be dismissed as "bad data" or not. Note that the Gaussian is centered on $MI_{\text{tot}} = 5.5865$. The histogram has been manually checked, and integrates to this same value. That is might be the case is not obvious in this semi-log plot, but the very high frequency of values $MI < 5$ is enough to balance the long tail on the right.

Scatter-plots

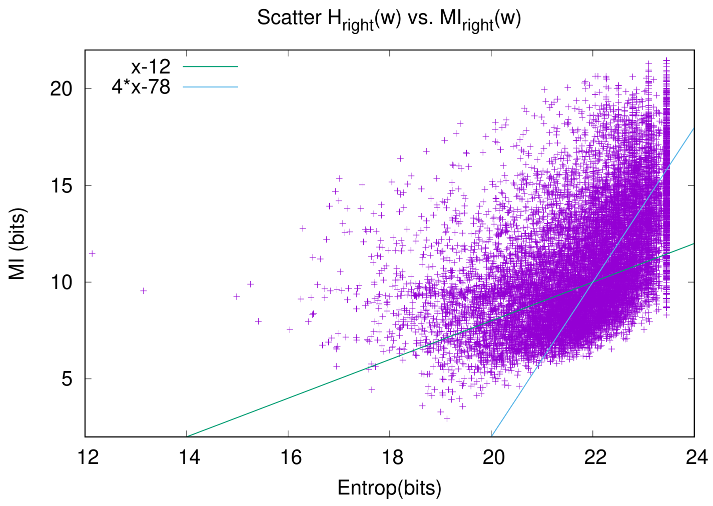
How does the MI correlate with the entropy and the frequency? Three scatter-plots reveal this relationship. The first scatter-plot shows the entropy $H_{\text{right}}(w)$ vs. the log of the marginal frequency $-\log_2 P(w,*)$. The two straight lines have the slopes as indicated, and approximately bound the distribution.



The graph below shows the mutual information $MI_{\text{right}}(w)$ vs. the log of the marginal frequency $-\log_2 P(w,*)$. Again, two lines approximately bound the distribution. Note that the average MI of this distribution is $MI_{\text{tot}} = 5.5865$. Thus, we have a very long tail offsetting a very heavy low-MI, high-frequency stub. Similarly, the average of $-\log_2 P(w,*)$ is $S_{\text{right}} = 8.5148$. Thus, the centroid of this scatter-plot is in the lower-left, and not where one might visually expect it.



Finally, the the entropy $H_{\text{right}}(w)$ vs. he mutual information $MI_{\text{right}}(w)$. Rather than attempting to bound this distribution, the two lines attempt to bisect it. This is an “unweighted” bisection. The actual centroid of this figure is at $H_{\text{tot}} = 19.463$ and $MI_{\text{tot}} = 5.5865$, at the lower middle of this scatter-plot.



In conclusion: there is an obvious correlation between frequency and entropy, and frequency and MI. However, the correlation between entropy and MI is considerably weaker.

Alternative Rankings

The Diary Part Three proposes that two words should form the seed of a cluster, if they have the largest ranked-MI of all word-pairs. Can the above provide some alternative rankings? First, recall the definition of the ranked-MI.

Ranked-MI

The following is a quick recap of the ranked-MI, as discovered and developed in Parts Three and Four of the Diary. The ranked-MI was built out of the common-MI plus Q, as follows. Specifically, this repeats and condenses the content of Part Three, pages 18-19, as well as Diary Part Four, page 19. Start by defining $N(w,d)$ as the observation count on the word-disjunct pair (w,d). From this, the observation frequency follows as

$$P(w,d)=\frac{N(w,d)}{N(*,*)}$$

The joint frequency of two words w,u is then defined as the matrix product

$$f(w,u)=\sum_{d}P(w,d)P(u,d)$$

This is not normalized, in that $f(*,*)\neq1$ which leads to the definition of

$$Q=-\log_2 f(*,*)=-\log_2\sum_{d}P(*,d)P(*,d)$$

which has a value of $Q = 11.945777$ for this dataset. Note the minus sign. This means $1 \gg f(*,*)$. The marginals can then be derived as

$$f(w) = f(w,*) = \sum_d P(w,d)\, P(*,d)$$

The word-pair MI is defined as

$$MI(u,w) = \log_2 \frac{f(w,u)\, f(*,*)}{f(w)\, f(u)}$$

Note the definition above is "projective", in that it is independent of the overall normalization of f. This is seen explicitly, by expanding out:

$$\begin{aligned} MI(u,w) =& \log_2 \frac{\left[\sum_d P(u,d)\, P(w,d)\right] \left[\sum_d P(*,d)\, P(*,d)\right]}{\left[\sum_d P(u,d)\, P(*,d)\right] \left[\sum_d P(w,d)\, P(*,d)\right]} \\ =& \log_2 \frac{\left[\sum_d N(u,d)\, N(w,d)\right] \left[\sum_d N(*,d)\, N(*,d)\right]}{\left[\sum_d N(u,d)\, N(*,d)\right] \left[\sum_d N(w,d)\, N(*,d)\right]} \end{aligned}$$

The ranked-MI is defined as

$$\begin{aligned} \text{ranked}MI(w,u) =& MI(u,w) + \frac{1}{2}\log_2 f(u) + \frac{1}{2}\log_2 f(w) + Q \\ =& \log_2 \frac{f(w,u)}{\sqrt{f(w)\, f(u)}} \end{aligned}$$

Just like the MI, the ranked-MI is scale invariant; its projective. This suggests that it has some interesting theory behind it. Graphs for the ranked-MI for this dataset can be found at the end of Part Three. They are pretty Gaussians.

Probabilities over Word Pairs

The ratio

$$\pi(u,w) = \frac{f(w,u)}{f(*,*)}$$

is a probability. Thus, we can play the game again, this time with π and ask about conditional probabilities, mutual information and entropies. For example, what is the value of

$$RMI_{\text{tot}} = \sum_{w,u} \pi(w,u) \log_2 \frac{\pi(w,u)}{\sqrt{\pi(w)\, \pi(u)}}$$

This and related questions have to stay out of reach, however, since computing $\pi(u,w)$ for all word-pairs (u,w) is computationally infeasible at this time. (Well, it could be done "easily" on a cloud instance with a few hundred CPU's, but I don't have that, or even the code to leverage that.)

Alternative Rankings

Following the above, several variations suggest themselves:

$$HMI\left(u,w\right)=MI\left(u,w\right)+\frac{1}{2}H_{\text{right}}\left(u\right)+\frac{1}{2}H_{\text{right}}\left(w\right)$$

and

$$MMI\left(u,w\right)=MI\left(u,w\right)+\frac{1}{2}MI_{\text{right}}\left(u\right)+\frac{1}{2}MI_{\text{right}}\left(w\right)$$

Either of these feel like promising improvements to ranked-MI, as, experimentally, ranked-MI is proving to be less than reliable for merge determinations. As mentioned earlier, the core problem is that ranked-MI seems to be suggesting that clusters should be merged far too often, presumably because $\frac{1}{2}\log_2 f\left(w\right)$ gets larger and larger as the clusters grow.

Neither of the above have "pretty" interpretations, which can be seen by recalling the definitions above:

$$H_{\text{right}}\left(w\right)=-\frac{\sum_d P\left(w,d\right)\log_2 P\left(w,d\right)}{P\left(w,*\right)}$$

and

$$MI_{\text{right}}\left(w\right)=\frac{1}{P\left(w,*\right)}\sum_d P\left(w,d\right)\log_2\frac{P\left(w,d\right)}{P\left(*,d\right)P\left(w,*\right)}$$

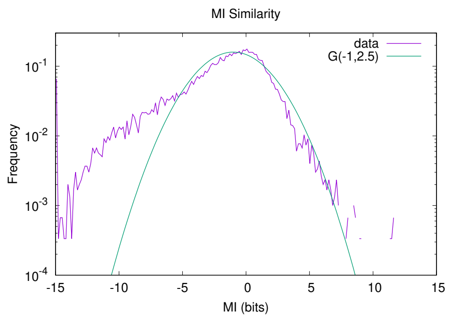
These are less compelling because of this complexity.

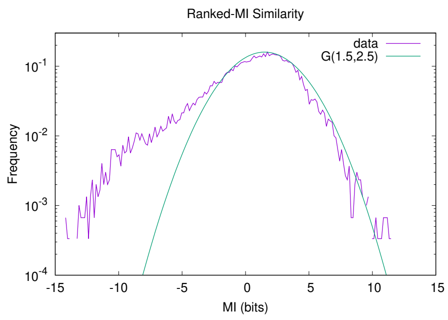
Alternative Similarity Distributions

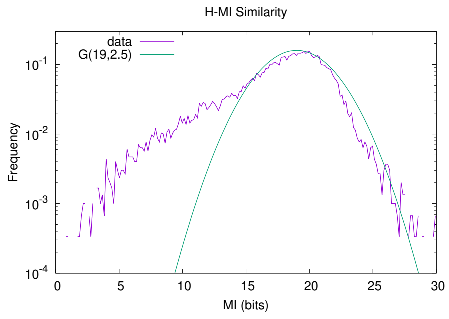
We previously saw that common-MI had a nice Gaussian distribution. Is this still the case for HMI and MMI? Lets find out ...

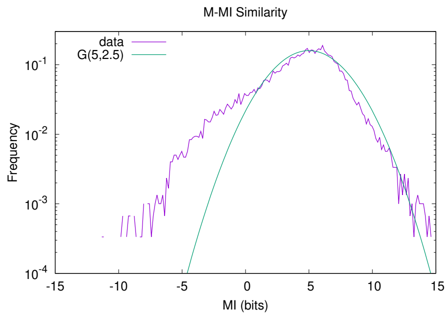
In Part Three, the graphs were done with a set of similarities for the top 1200 words, computed without shapes. What I have handy for the current dataset is much smaller: just the top 200 most frequent words; however, this time, similarities were computed using shapes. So its .. different. Anyway, there's a total of $N\left(N+1\right)/2=20100$ similarities handy. Of these, 175 do not have an MI; that is, the dot-product is zero, so the MI is $-\infty$. Thus, only 20100-175=19925 actual pairs.

Four nearly identical graphs, differing only in the horizontal offset:



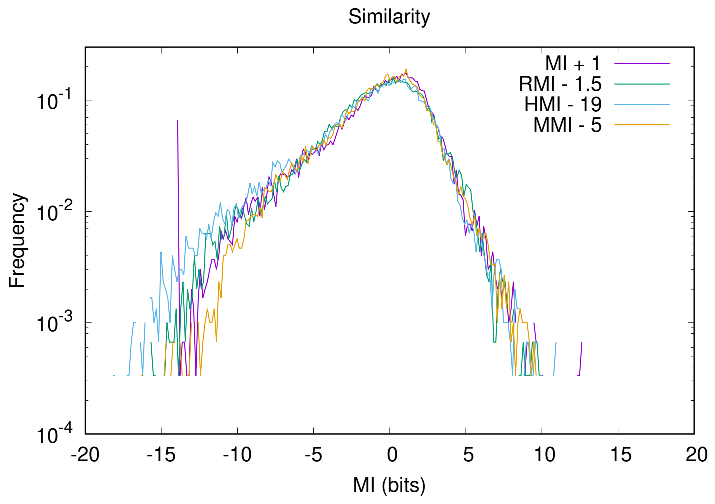






The Gaussians are an eyeballed fit. Perhaps there are some popular fat-tailed distributions that would give a better fit? At any rate, the fat-tail is on the left, an excessive of word-pairs judged to be dis-similar. Perhaps this is a good thing? That is, the MPG parsing gave us a bunch of disjuncts that are "obviously" dis-similar, which is exactly what we hope to see. Perhaps the fatness of the tail is correlated with the number of distinct word-senses? That is, we expect words with different word-senses to be dis-similar to one-another, and so the more word-senses there are, the fatter the tail? What is the theoretical exposition for this effect?

How similar are these graphs? All four are re-plotted below, with the horizontal offset adjusted by the eyeballed fits, above.



Clearly, these are almost identical, but not quite. Given the earlier scatter-plots, the specific rankings of specific pairs must be different in each of these. How different? The table below lists the top 17 most similar pairs for each style of similarity. The top number is the similarity score for each style; the negative numbers are how much less the next score is, from the top score.

MI		ranked-MI		HMI		MMI	
9.755	[+	9.167	— +	28.02	— +	19.32	[+
-0.37	— +	-0.01	; ,	-0.01	[+	-1.58	— +
-1.69	— [	-0.11	is was	-0.44	— [	-3.02	— [
-2.63	! ?	-0.14	and but	-0.76	their our	-3.21	(+
-2.79	(+	-0.23	. ?	-1.41	their its	-3.53	] +
-3.42	] :	-0.25	! ?	-1.49	your our	-4.05	([
-3.45	] +	-0.30	It He	-1.64	their your	-4.31	] [
-3.46	should might	-0.35	[+	-1.64	! ?	-4.91	] :
-3.51	([	-0.94	” "	-1.77	their his	-5.02	* +
-3.82	should could	-0.97	No A	-1.78	make take	-5.55	! ?
-3.90	their our	-0.97	in of	-1.81	its our	-5.70	— (
-3.91	. ?	-0.97	she he	-1.88	make get	-5.88	] —
-3.95	— (	-1.02	It There	-1.93	upon into	-6.18	[*
-3.96	were are	-1.06	and as	-1.98	his our	-6.48	] (
-3.99	upon on	-1.10	! .	-2.00	my your	-6.65	)]
-4.03	might could	-1.13	‘ “	-2.06	upon from	-6.83	should might
-4.06	in into	-1.20	She It	-2.07	get take	-7.31	well long

What can we conclude? Clearly, the naked MI score is exploring all pairings of similar things. By using the in-group algo, we could expect that all the punctuation gets lumped together. The HMI seems to fall into a similar trap: the in-group seems to be “their our its your his my”.

Hard to figure out what to make of this.

A tempting conclusion, perhaps a bit *ad hoc*, is to use ranked-MI to suggest the initial pair to seed a cluster, but then to use either MI, HMI or MMI to expand the actual in-group. So, MMI looks pretty slick, but is computationally the heaviest to recompute. HMI should be fairly fast, as it’s just some logs. And MI is computationally “free”, it has to be computed anyway.

Where to change this: search code for ‘optimal-in-group‘ and specify the desired similarity function.

Hypothesis: it is also possible that these different in-group suggestions won’t alter very much. This is because it is not the initial pair that matters, it is the in-group near that initial pair that matters. Perhaps all of these end up suggesting similar, or the same in-groups?

Experiment-11 – Deep Merge (also Expt 12 graphs)

Given the above results, it seems time for a deeper-merge experiment. Unlike experiment-10, this one focuses on:

- Using plain-MI instead of rank-MI for determining the in-group.
- Fixes quorum=0.7 and commonality=0.2 and varies the noise: noise= 4,3,2,1

Based on expt-10 it looks like quorum=0.7 or 0.8 would give better results. Also, it seems like commonality has almost no effect: the chosen grouping pretty much always is arrived at with the back-off strategy, rather than with commonality exceeding the threshold. By contrast, the noise-threshold appears to have a huge effect on what gets merged. Here's an example. For the very first merge, we see

```
------ Round 1 Next in line:
ranked-MI = 9.1671 MI = 9.3888 (‘‘, ‘+‘)
------ Start merge 1 with seed pair ‘‘ and ‘+‘ Initial in-group size=8: ‘+‘ ‘‘ ‘‘ ‘_‘ ‘)‘ ‘[‘ ‘(‘ ‘]‘
In-group size=8 overlap = 9 of 10975 disjuncts, commonality= 0.08%
In-group size=7 overlap = 17 of 10530 disjuncts, commonality= 0.16%
In-group size=6 overlap = 33 of 10107 disjuncts, commonality= 0.33%
In-group size=5 overlap = 29 of 9433 disjuncts, commonality= 0.31%
In-group size=6: ‘+‘ ‘‘ ‘‘ ‘_‘ ‘)‘ ‘[‘
```

Notable is just how pathetically low the commonality is. However, when the merge is actually performed, we see something quite different. For noise=4, we get

```
------ merge-majority: Merge 5550 of 7957 sections in 12 secs
------ merge-majority: Remaining 14284 of 22825 cross in 43 secs
```

which tells us that almost all of the merged sections had an observation count of less than or equal to 4! That is, 5550+14284 were merged, whereas we expected only 6 x 33 x 70% = 140 to get merged! Yowie! This means that only 140 sections/cross-sections had an observation count of greater than 4; the rest of them had a lesser count, and were merged only because they were swept up as being under the noise floor.

For noise=3, we see that the number of merged sections drops:

```
------ merge-majority: Merge 4884 of 7957 sections in 11 secs
------ merge-majority: Remaining 12502 of 22863 cross in 37 secs
```

For noise=2, the number of merged sections drops again. It's tempting to say that it's "surprisingly high", but its not: The Zipfian distribution tells us that there's an incredibly fat tail at these low counts.

```
------ merge-majority: Merge 3586 of 7957 sections in 10 secs
------ merge-majority: Remaining 9135 of 22963 cross in 35 secs
```

How much harm is there in merging of these low-count tails? My current guess is "none". These low-count tails contribute very little, or nothing to the word-pair MI. Right? Or are they nickel-n-diming us to death? (Hmm. If they contribute nothing, then they could have been trimmed away from the get-go...) Mostly, they hang around, chewing up space... Later on, when a dictionary is created, these can be trimmed away, to arrive at a more compact dictionary. I don't have insight here. It's confusing.

On the other hand, due to the preferential-attachment phenomenon that is being seen (large clusters getting larger) it might be that these fat tails are wagging the dog:

they suck in the noise, and then this noise sucks in other tails. These tails might end up dominating the sense of the cluster? Perhaps we should have a noise=0 run to compare against?

At any rate, its kind of surprising that the MI can be high but the commonality is so low. This doesn't make sense intuitively.

What contributes to the MI?

The last sentence immediately above seems important enough to get it's own subsection. What, exactly, is contributing to the MI? We see high MI values, but a low Jaccard similarity. This seems surprising. Is it because the MI's are pairwise, while the Jaccard is for the group as a whole?

How does the low-count tail influence the MI? When does trimming tails help? When does it hurt?

Progress graphs

Far more graphs from this run. All of these graphs are for datasets for which quorum=0.7 and commonality=0.2. They differ in the noise setting, varying from 4 to 1. The "older" dataset is from the previous run, way up above, which had quorum=0.7, commonality=0.4 and noise=4. Note that assorted buglets were fixed since that older run.

The "precise" dataset recomputes all word-pair MI for every possible word that was affected by the merge. This can be hundreds of words, thousands of word-pairs (and is thus slow to compute). The "imprecise" datasets recompute the MI only for the words directly involved in the merge (so, for only a handful of words, and so much faster to compute.) Thus, one of the questions is whether the "precise" datasets provide materially better classifications, or not.

Computation of these datasets was terminated with a snowstorm power outage. Rather than resuming where these left off, a fresh computation was started, with a slightly cleaned up dataset, Experiment 12. As can be seen, there are no substantial changes arising from this.

Shown for comparison are the Experiment-12 graphs. These are based on the same underlying dataset as Experiment-11, but are cleaned up to remove unconnectable words and disjuncts. That is, the dataset is scrubbed to remove all connectors that cannot connect to words, and vice-versa. This has a minor effect on the graphs, but reduces the amount of annoyance and confusion during inspection and debugging.

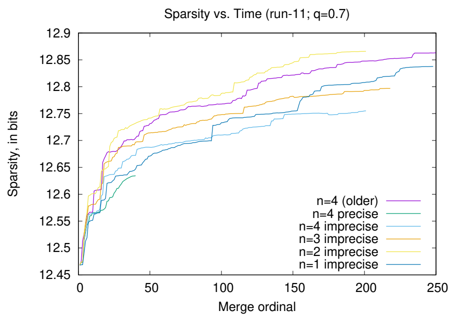
The Experiment-12 "imprecise" runs all failed when they ran out of hugepage memory. Foo. Unable to recover and pick up where they left off; there is some kind of bug that prevents restarting (the marginals get unbalanced, and recomputing them leaves something wrong...) These halted as shown:

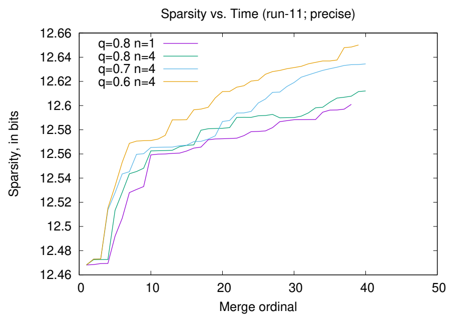
Dataset	nmerges
r12-log-q0.7-c0.2-n1.dat	1343
r12-log-q0.7-c0.2-n2.dat	891
r12-log-q0.7-c0.2-n3.dat	1062
r12-log-q0.7-c0.2-n4.dat	1058

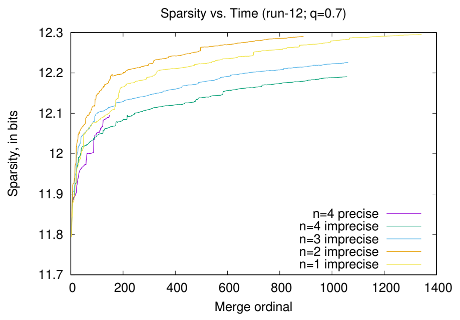
The last two stopped at almost the same place, and thus allow head-to-head comparisons. See much farther below.

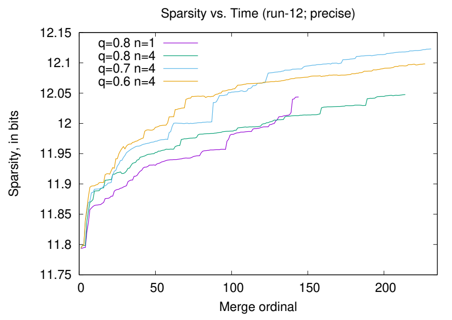
Sparsity

Sparsity is defined as always: it is log2 of the fraction of non-zero matrix entries to total possible matrix entries. Gut instinct says 'slower sparsity growth is better'. From the first graph, though, we find a very non-linear growth rate depending on the noise threshold.









For the new improved cleaned-up dataset, the initial sparsity is lower.

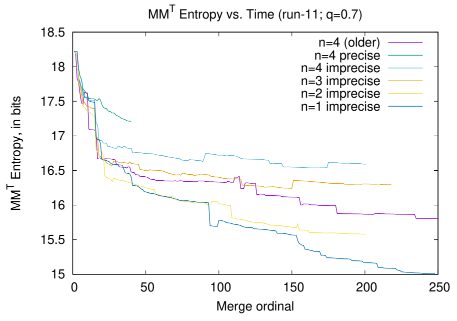
MMT Entropy

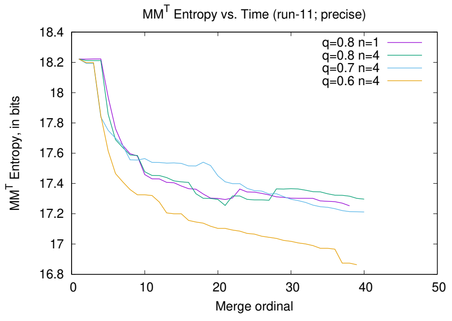
The MM^T-entropy is as defined before; it is

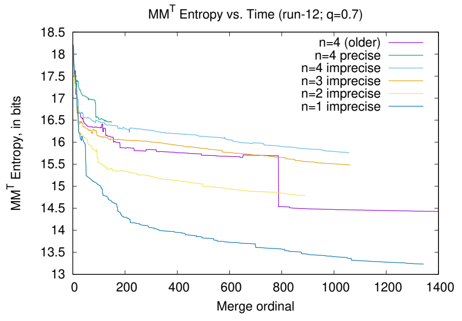
$$H = \log_2$$

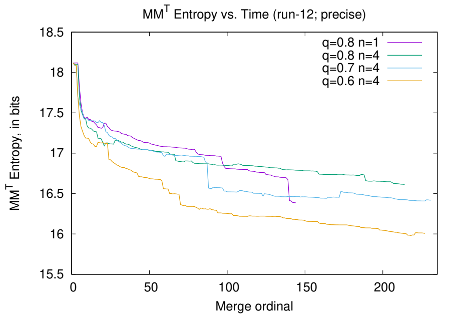
where

$$f(w,u) = \sum_d P(w,d)\, P(u,d)$$









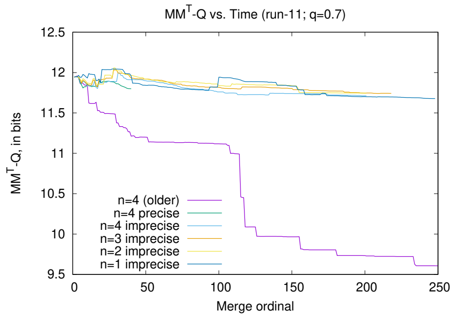
Note that the cleanup has almost no impact on the MMT Entropy. It is slightly different, but not by much.

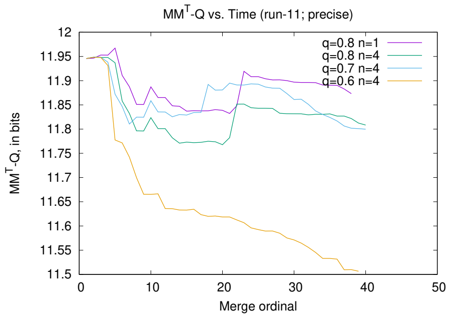
MMT-q

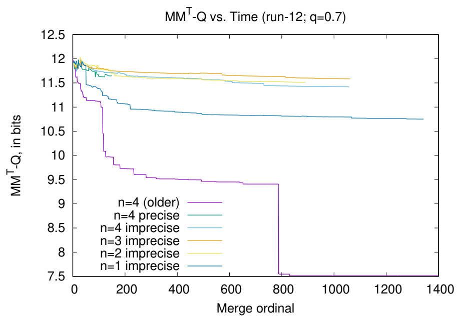
The $MM^T q$ is the Q value as defined before, above, and in 'Diary Part Four':

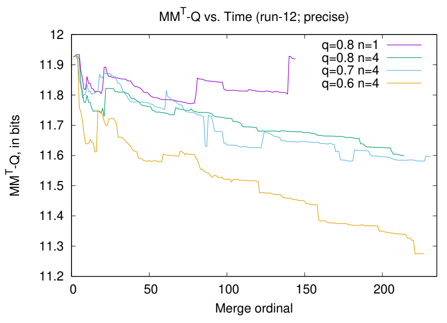
$$Q = -\log_2 \sum_d P(*,d) P(*,d)$$

It varies over time, as both new disjuncts are created, old disjuncts have counts drop to zero, and similarly, come words have counts drop to zero as they are transferred to the word-classes.





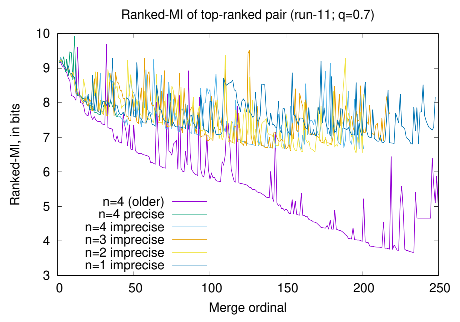


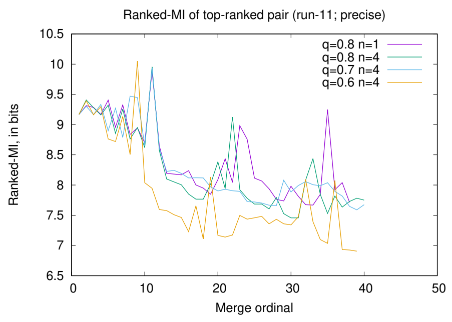


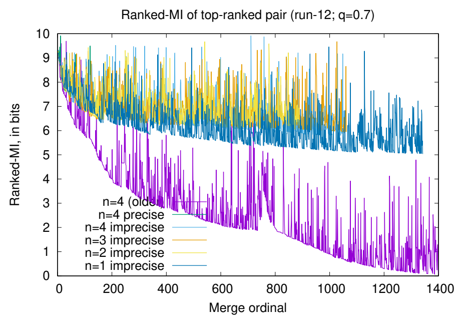
Ranked-MI

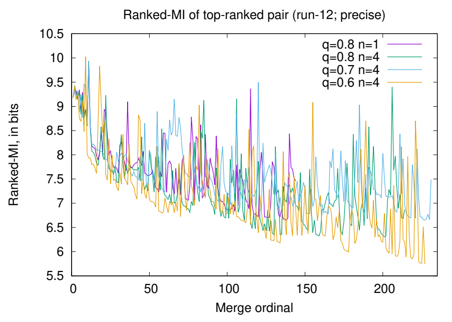
The next graph shows the ranked-MI of the top-ranked word-pair that triggers the merge. Were it not for the re-computations of word-pair MI's, this should trend downwards, monotonically. The re-computation of MI exposes new fragments/remainders of merged classes that elevate this MI, briefly, until undisturbed word-pairs are encountered again.

Ranked-MI is defined as in 'Diary Part Four', and restated earlier, above.



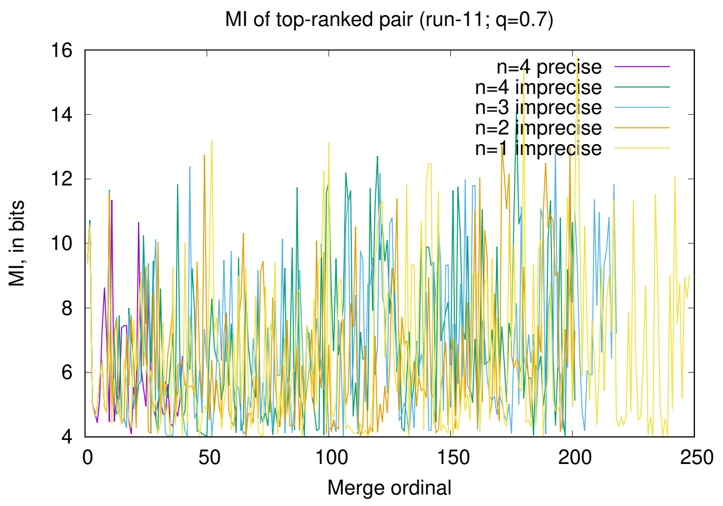






MI

The ranked-MI is build as a linear combination of the word-pair MI, plus the log of the frequencies, plus Q. It seems reasonable to ask what the MI was of the pair that initiated the merge. This is shown below.



Marginal Entropies

The next four graphs require the frequencies $P(w,d)$ to be computed. This turns out to be error-prone and challenging, and thus will be reported here, but turned off in future versions of the code (can be re-enabled by setting (define TRACK-ENTROPY #t) in the code.). The problems are these:

- The pair-frequencies must be recomputed before starting, and stored. This includes the frequencies on CrossSections, which are not normally stored. After this, the marginal frequencies and the marginal entropies must be computed.
- As the merges proceed, entropies must be recomputed (they are, if the flag is set). This adds approx 5 minutes to each merge cycle, which is a significant overhead.
- The alternative is to create a new matrix API for entropies that always computes what is needed on-the-fly, from the counts (which are always correct). Doing this is ... OK, as it will reduce operational errors, but will add yet more CPU overhead to obtaining these stats.

Given these issues, it seems that tracking entropies is not worth it. I mean, the graphs below are interesting, and all, but they are not Earth-shaking, and don't seem to offer any new insight, beyond some warm fuzzies that everything is going fine.

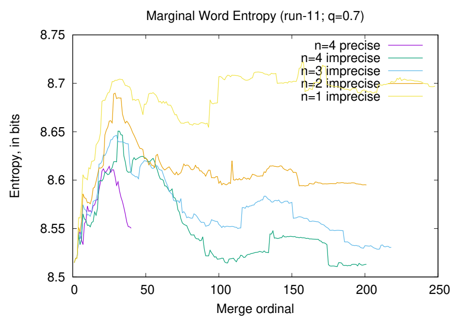
These remarks apply to some of the class entropies, much further below. They won't be available because of the dependency on the pair-freq-api object.

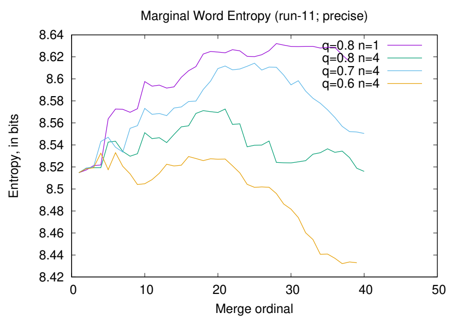
Marginal Word Entropy

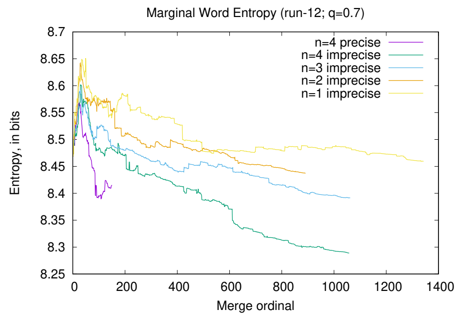
The right-marginal or word entropy is as defined above:

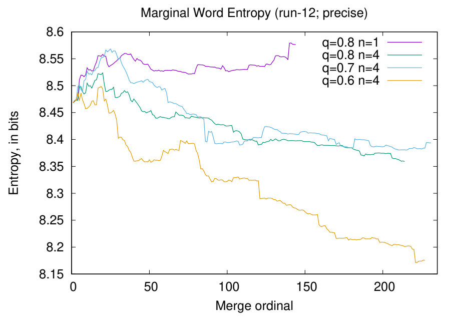
$$S_{\text{right}} = \sum_{w} P(w,*) \log_2 P(w,*)$$

Yes, this probably should have been called “left”, but that introduces new confusions.





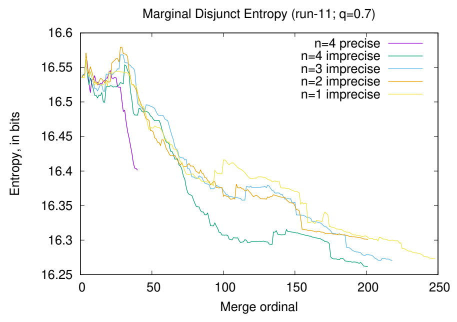


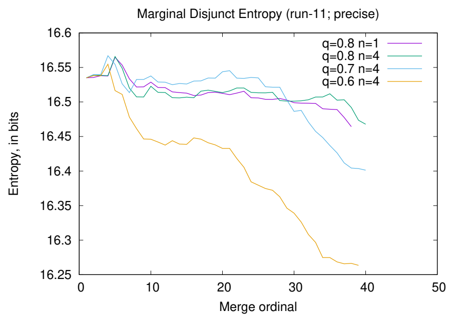


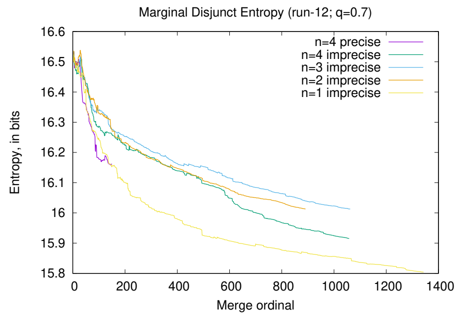
Marginal disjunct entropy

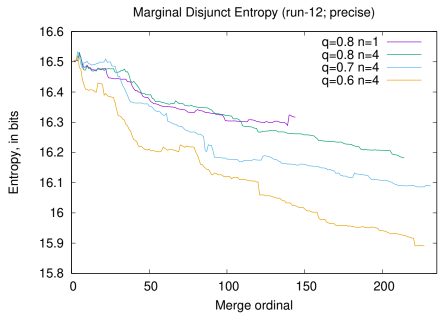
As defined above:

$$S_{\text{left}} = \sum_d P(*,d) \log_2 P(*,d)$$





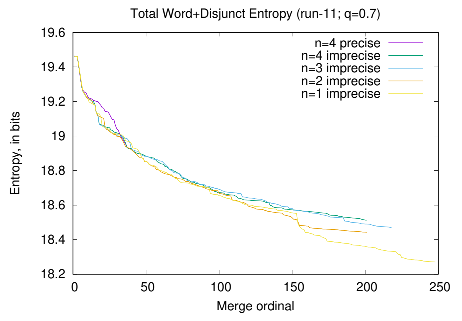


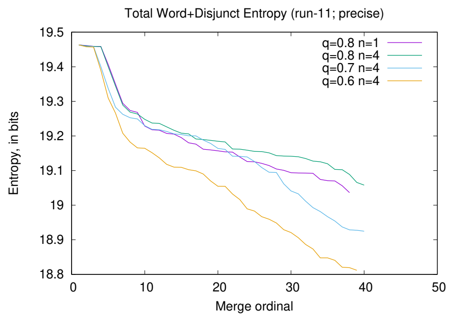


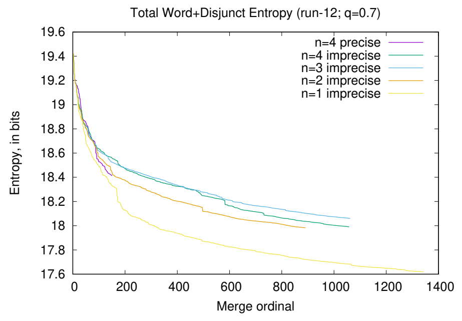
Total Entropy

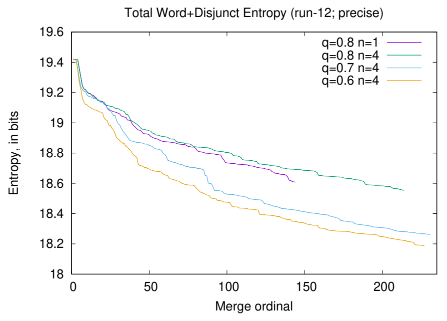
The total entropy is as defined above

$$H_{\text{tot}} = -\sum_{w,d} P(w,d) \log_2 P(w,d)$$





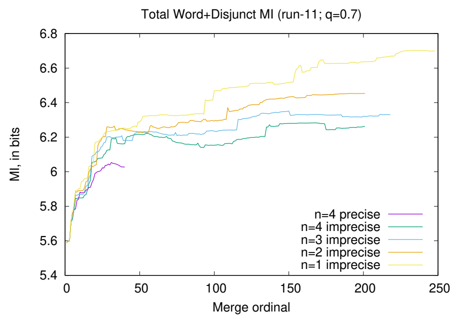


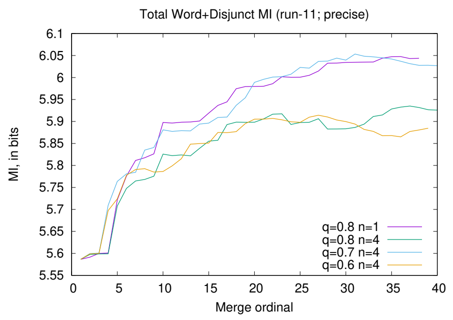


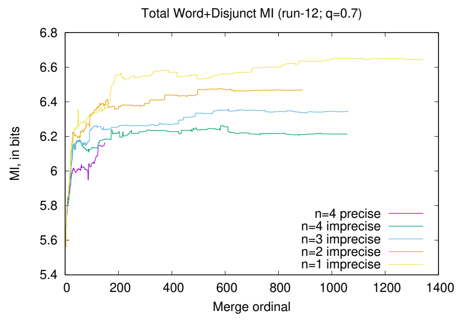
Total MI

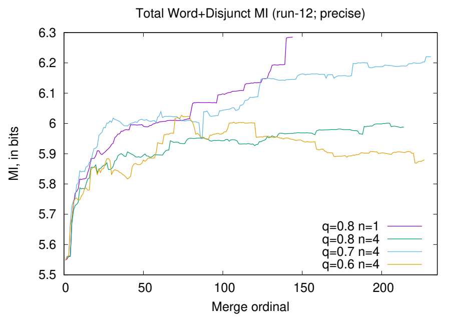
The total MI is as defined above:

$$\begin{aligned} MI_{\text{tot}} =& \sum_{w,d} P(w,d) \log_2 \frac{P(w,d)}{P(*,d)\,P(w,*)} \\ =& S_{\text{right}} + S_{\text{left}} - H_{\text{tot}} \end{aligned}$$



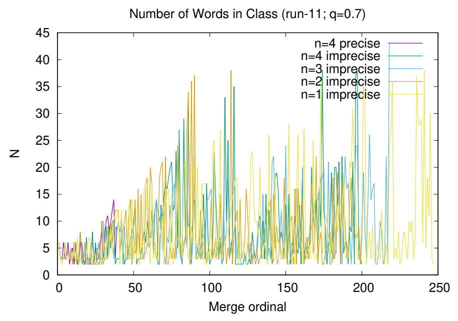


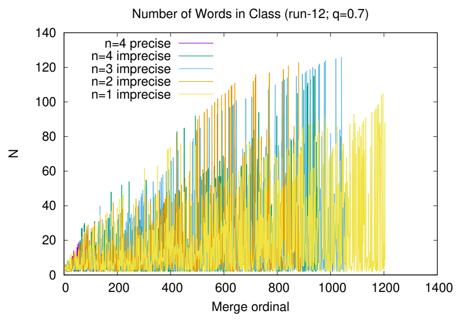




Number of Words per Class

As mergers proceed, it is not uncommon to lump additional words into an existing class (possibly splitting that class into two). Thus, over time, one expects the size of classes to grow. This is charted below. It shows the size of the word-class created at that time-step.





Self-MI of a Class

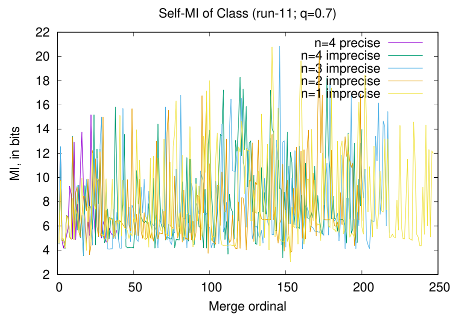
The MI of two words is as defined above:

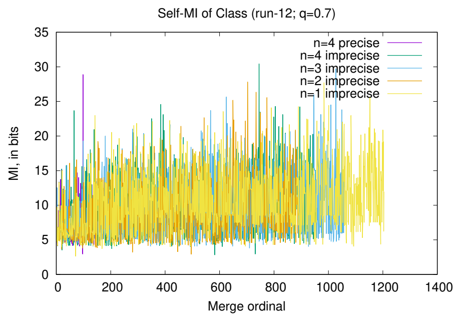
$$MI(u,w) = \log_2 \frac{f(w,u)\, f(*,*)}{f(w)\, f(u)}$$

where

$$f\left(w,u\right)=\sum_{d}P\left(w,d\right)P\left(u,d\right)$$

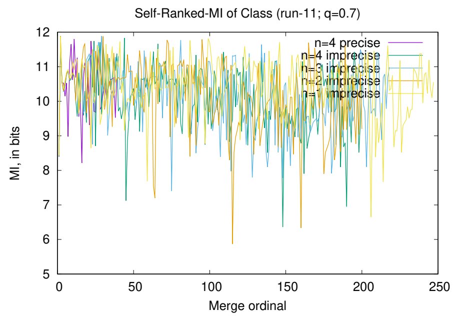
The self-MI of a word w is them $MI\left(w,w\right)$. The graph below shows the self-MI of the class created at that time-step.

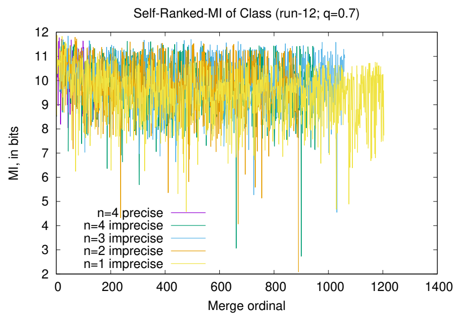




Self-ranked-MI

As above, but this is the ranked-MI of the class against itself.





Support of the Class

The support of a word is the grand-total number of disjuncts assigned to that word; likewise for a word-class. In formulas, the support is

$$\Delta\left(w\right)=\sum_{d}\delta\left(w,d\right)$$

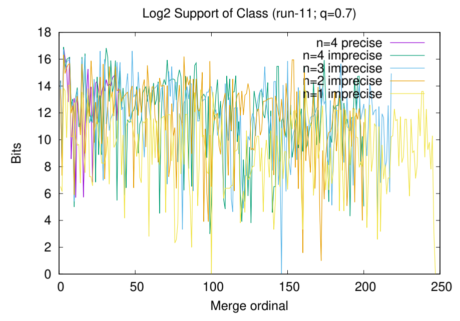
where

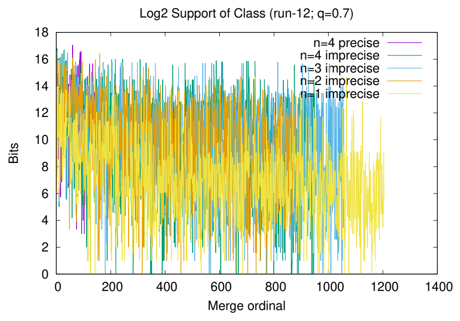
$$\delta\left(w,d\right)=\begin{cases}1 & \text{if } N\left(w,d\right)>0\\ 0 & \text{otherwise}\end{cases}$$

Shown below is $\log_{2}\Delta\left(w\right)$ for $w$the recently created class. Thus, the larger this is, the greater the support.

It might be more edifying to subtract this from the total number of disjuncts: this is mildly variable, approximately equal to 1.05×10^{6} – that is, a little over a million disjuncts, or about $\log_2 1.05\times10^{6}\approx 20$. This would then give the fraction of all possible disjuncts appearing in the given word-class.

It might be interesting to relate this to the size of the word-class.



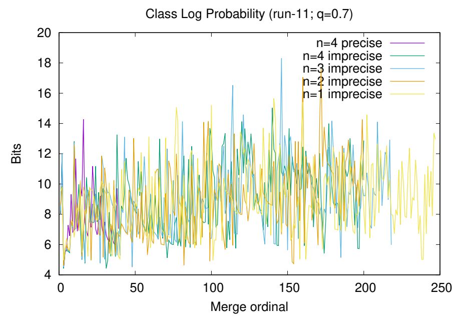


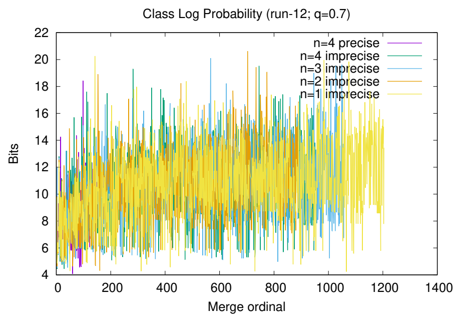
Marginal Probability of the Class

The marginal probability of the class is as defined above:

$$P(w) = P(w,*) = \sum_d P(w,d)$$

The graph below shows $\log_2 P(w)$. It might be interesting to look at how this correlates with the support.





Class Fractional Entropy

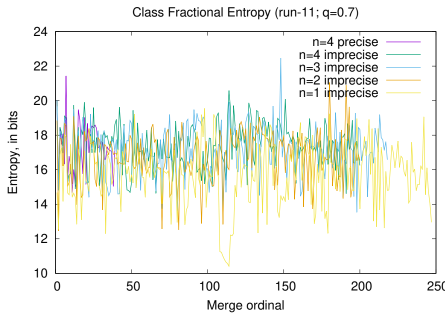
The entropy of a word is as defined earlier:

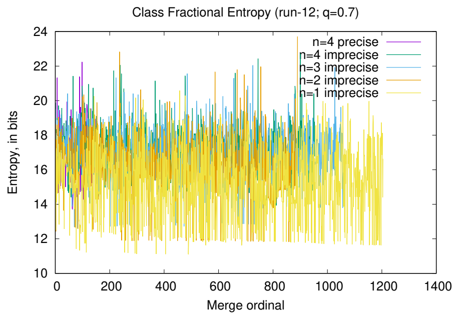
$$h(w) = -\sum_d P(w,d)\log_2 P(w,d)$$

The fractional entropy is

$$H(w) = \frac{h(w)}{P(w,*)}$$

This is graphed below.





Compositional Entropy

The compositional entropy is a brand-new concept not discussed earlier in this diary. It is the entropy of the of the words that went in to form the class.

When a class is formed out of the words $w_1, w_2, \cdots, w_n$, the total counts of each of these contributions $N(w_1,*), N(w_2,*), \cdots, N(w_n,*)$ is tracked on the MemberLink for that class. Define the total observation count on this class as

$$N_C = \sum_k N(w_k,*)$$

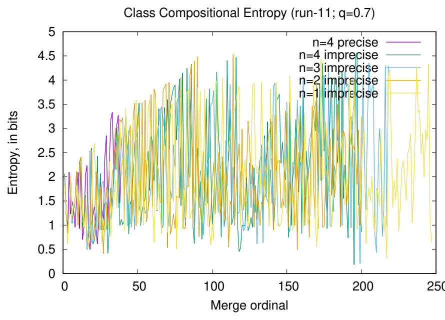
This can be used to define a class-relative fraction of how much each word contributed to the class:

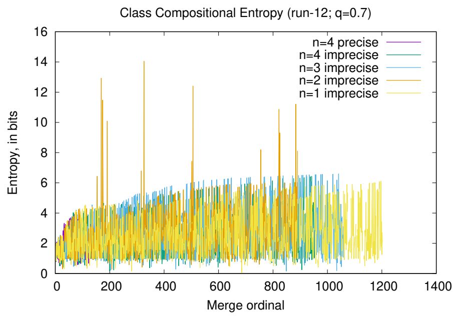
$$P_C(w_k,*) = \frac{N(w_k,*)}{N_C}$$

The compositional entropy is then the ordinary entropy with regards to this compositional probability. It is defined as

$$CH = -\sum_k P_C(w_k,*) \log_2 P_C(w_k,*)$$

It is shown in the graph below.





The compositional entropy is interesting because it gives a hint of how different words are contributing to form the class: it's a weighted average of the contributions. For example, for two words, the compositional entropy is maximized when both words contribute equally (and is equal to 1.0). Given how the above is scaling, it might be interesting to redo this graph, dividing by (the log of) the number of words in the class.

Experiment-12

Some ongoing confusion motivates an attempt at a cleaner starting point for merges. Start with ‘run-1-t1234-tsup-1-1-1.rdb‘ and build a new, clean file, using the scripts in ‘scm/gram-class/cleanup.scm‘. Here’s where it’s at:

	r1-t1234-tsup-1-1-1	r12-mst.rdb	r12-shape.rdb
N_L= words	15083	9495	9495
N_R= dj	205003	204680	1015850
D_{Tot}= size	855718	833833	2717117
N_{Tot}= obs	7313131	7202301	22643824
$\log_2 N_{\text{word}}$	13.881	13.213	13.213
$\log_2 N_{\text{dj}}$	17.645	17.643	19.954
$\log_2 D_{\text{Tot}}$	19.707	19.669	21.374
sparsity	11.819	11.187	11.794
rarity	3.944	4.241	4.790
$\log_2 N_{\text{Tot}}/D_{\text{Tot}}$	3.095	3.111	3.059
Entropy	16.352	15.720	18.117

Recall that

$$\text{Sparsity} = -\log_2 D_{\text{Tot}}/N_L \times N_R$$

while

$$\text{Rarity} = \log_2 D_{\text{Tot}}/\sqrt{N_L \times N_R}$$

Here’s the word-disjunct entropies and MI’s; its in r12-mi.rdb which is just ‘r12-shape.rdb‘ with the MI scores in it.

name	Entropy total	Left	Right	MI
r12-mi.rdb	19.421	16.501	8.4683	5.5480

What’s done:

- ‘r12-mst.rdb‘ copies ‘run-1-t1234-tsup-1-1-1.rdb‘ and forces bi-directional linkage symmetry. It also recomputes support marginals and MM^T via contents of ‘run-common/marginals-mst.scm‘.
- ‘r12-shape.rdb‘ adds shapes, and recomputed the MMT marginals for these. The dimensions includes the CrossSections.
- ‘r12-mi.rdb‘ batch-computes the word-disjunct MI (with ‘batch-all-pair-mi‘).
- ‘r12-mi-sim200.rdb‘ is just ‘r12-mi.rdb‘ with word-similarities pre-computed for the top 200 ranked words.

The Crash

As noted above, the runs crashed. Will try restarting one more time, after nuking invalid CrossSections. Nope. That doesn't work.

Similarity between WordClasses vs. Words

Two datasets halted at almost the same place. Perhaps they are comparable. As below.

Dataset	nmerges	nclasses	nwords
r12-log-q0.7-c0.2-n3.dat	1062	536	9490
r12-log-q0.7-c0.2-n4.dat	1058	530	9486

In both cases, the number of word-classes that survived is almost exactly half the number of merges run (why half?) This is because classes get repeatedly merged into, instead of having new classes being created out of the blue. Kind of surprising, this needs to be understood better.[3]

Anyway: One hypothesis: the similarity distributions between word-classes should be low. That should be quite distinct from one another, one might hope. So, three batches of similarity distributions:

- Similarity between all of the 530 (or 536) word-classes. (not counting self-similarity).
- Similarity between the top-ranked 530 words.
- Similarity between the top-ranked words and word-classes.

What will it be like? Can we expect the third bullet to show higher similarity than the second?

All similarities will be recomputed from scratch, as the existing similarities are tainted by the altered disjuncts – i.e. the old similarities might not have been recomputed, after a change to their disjuncts. Only the "precise" merges recompute all similarities.

Conclusions – Summary

Lets summarize what was found below. First, the self-similarity distribution for word-classes and other words are dramatically different. Word-classes have a high self-similarity. The shapes of the distributions are different, too.

The distribution of non-zero similarities between classes, and classes and words, and just words, are all more-or-less normal distributions (Gaussians), all quite similar, all centered on a small negative MI (around -3 to -1.5, depending) and fairly narrow (a standard deviation of about 3). There are no dramatic differences at this level.

[3]The datasets for the below are located in the p5-class-sim directory. The scripts used to build the datasets are in the utils/similarity-p5.scm.

There is a very dramatic difference, when looking at the fraction of all possible pairs with zero similarity! Zero similarity corresponds to $-\infty$ in the MI value; that is, there is zero overlap between the disjuncts. They are orthogonal.

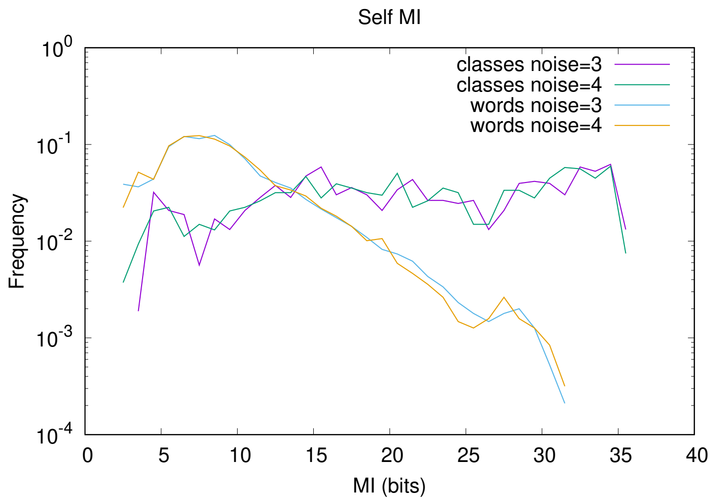
Of all of the possible similarities between the word-classes, only 7% are non-zero. The classes are highly orthogonal to one-another; only a few of them are not. By comparison, more than 80% of the words have non-zero similarity between one another. The words are mostly not orthogonal to one-another. In the middle are the similarities between words and classes, coming in at just 20% of all pairings.

Conclude: the classification algorithms is fairly good at "orthogonalizing". Post-classification, the classes are fairly distinct from one-another, quite unlike the situation with raw words. During classification, it appears that the algo is "sweeping up" words in an effective way: The 20% figure seems to say that classification really is vacuuming up the words.

The above suggests a strong signal for the quality of classification: the orthogonal fraction of pairings. All three of the above figures should be tracked during classification.

Self-similarity Distributions

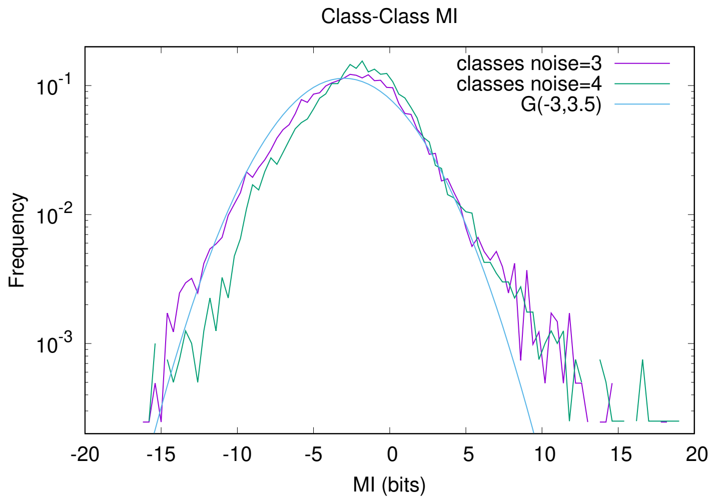
The similarity is given by the (symmetric) word-MI. The self-similarity is the self-MI. Here we go:



Above is the distribution of the self-MI for the 530 word-classes and the 9490 words in r12-log-q0.7-c0.2-n3.dat. Also, the 536 classes and 9486 words in r12-log-q0.7-c0.2-n4.dat. Clearly, seems to be quite different. What does it mean?

Class–Class Similarity Distributions

The noise=3 dataset had only 10154 non-vanishing similarities out of a total of $(530 \times 531)/2 = 140715$ possible pairings of word-classes with one-another (excluding self-similarity). For noise=4, there were only 9988 out a possible 536 x 537/2 = 143916 pairings. This means that only seven percent (approx.) of the word-classes are similar to each other enough to have at least one disjunct in common. The rest had zero disjuncts in common! So, word classes are not perfectly orthogonal to one-another, but most of them are, considered pair-wise. That's a pretty hot result. I like it.

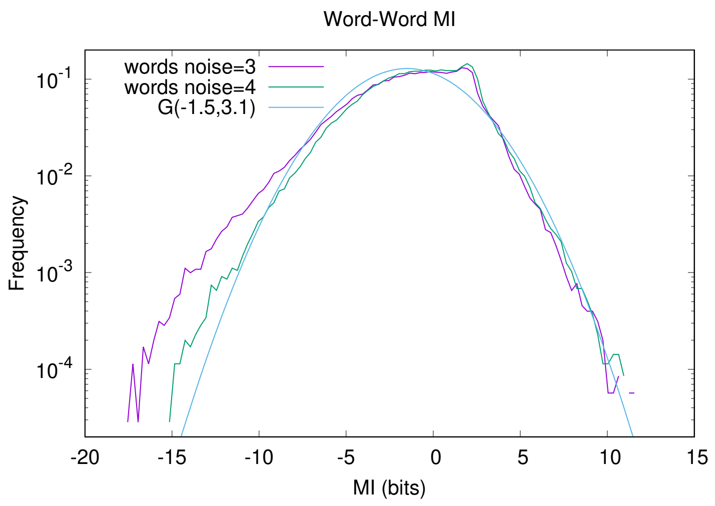


The Gaussian G(-3,3.5) is just an eyeballed fit, as usual. The noise=3 curve seems to be shifted slightly to the left. One would expect that better clustering shifts these curves leftward.

Word–Word Similarity Distributions

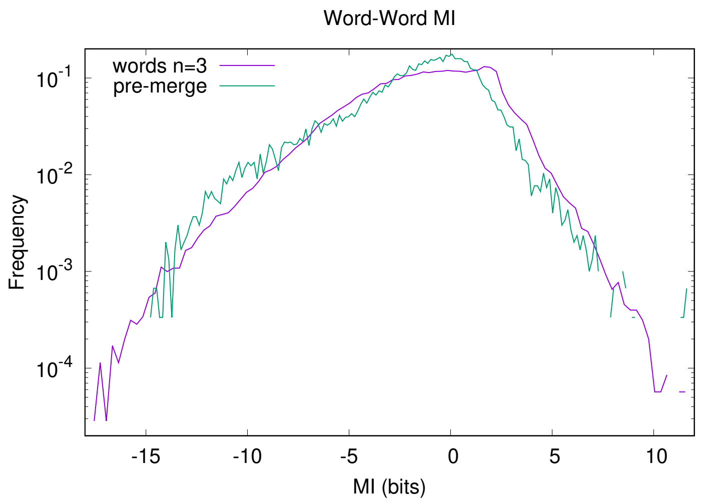
The noise=3 dataset had 117218 non-vanishing similarities out of a total of $(530 \times 531)/2 = 140715$ possible pairings of word-classes with one-another (excluding self-similarity). For noise=4, there were 117023 out a possible 536 x 537/2 = 143916 pairings. So this time, approx 83% of all possible similarity pairs have similarities. A very non-orthogonal situation.

The selected words were the top-ranked ones, out of all possible words. The number of words was limited, with the idea that they would be comparable in number to the classes.



The Gaussian G(-1.5,3.1) is just an eyeballed fit, as usual. Note that it is shifted to the right, and is narrower, than the Gaussian in the previous figure. The noise=3 curve seems to be shifted slightly to the left. It is not clear why it should be shifted.

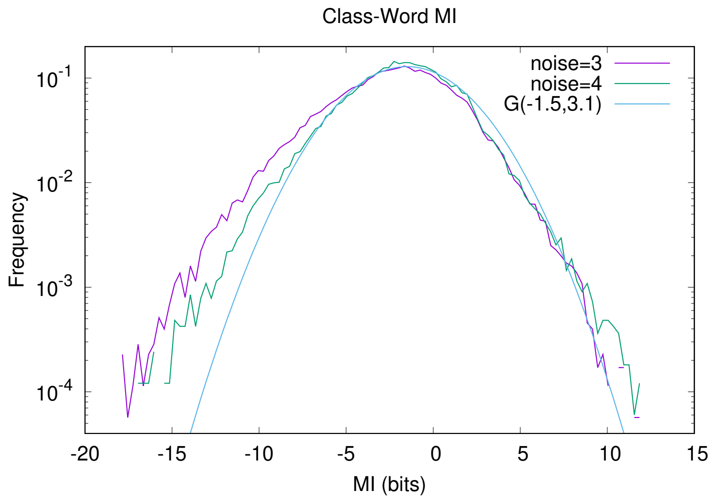
Compare this to the earlier, pre-merge word-word MI distribution graph, about half-way up in this document. It's the same distribution, but before any merges have been done. This is shown below, relative to the current noise=3 graph. Both the mean and the stddev are about the same; maybe a bit of a shift of the mean.



The "pre-merge" curve is the one from half-way up: it shows similarities for only 200 words, and so is noisier. The shape of the fat tail is ambiguous. Earlier, it looked like a well-defined Gaussian with a very fat tail; but here, this is not so clear. Clearly, noise=4 suppresses the tail a lot more than noise=3.

Class–Word Similarity Distributions

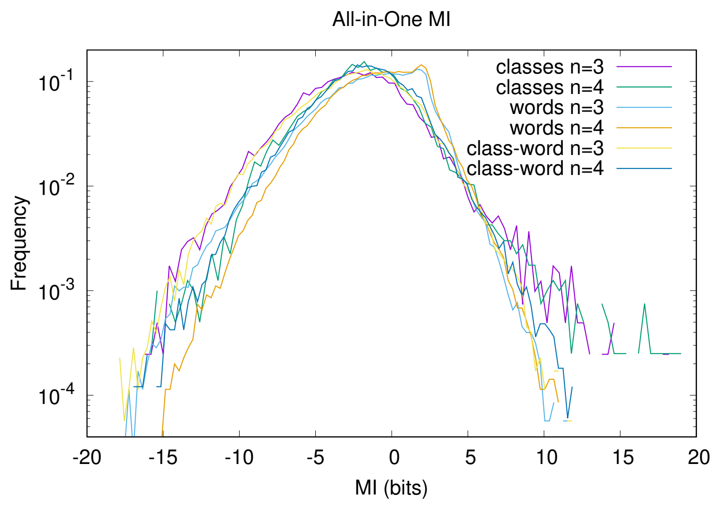
The noise=3 dataset had 58555 non-vanishing similarities out of a total of $530^2 = 280900$ possible pairings of word-classes with one-another (excluding self-similarity). For noise=4, there were 55194 out a possible $536^2 = 287296$ pairings. So this is a relatively meek 20% of all possible pairings. Conclude that the word-classes have swept up those things that have overlaps.



The Gaussian G(-1.5,3.1) is just an eyeballed fit, as usual. This is the same Gaussian as used above. The noise=3 curve seems to be shifted slightly to the left. It is not clear why it should be shifted.

All-in-one Similarity Distributions

The below reproduces all of the above curves, in one figure.



Hmm. Graphed this way, it seems that there are no dramatic differences. There are subtle differences, already noted.

To conclude: perhaps the most surprising is not the distributions of the similarities, but the fraction of the pairs that have non-zero similarity. This is the dramatic result we were looking for, even if its not evident in these graphs.

The End

This is the end of Part Five of the diary.